Research on Multiple-Valued Logic At The Naval Postgraduate School

Jon T. Butler
Department of Electrical and Computer Engineering
Naval Postgraduate School

Abstract

The Navy's need for faster, smaller computers has inspired research on the use of more than two levels of logic in computer circuits. This paper focusses on work at NPS in multiple-valued logic. It discusses the development of multiple-valued programmable logic arrays (MVL-PLA), as well as computer-aided design tools for MVL-PLA's.

Introduction

The effectiveness of Navy systems is dependent on computers. Computers are used to predict weather, to track resources, to guide missiles, to simulate combat, and to process the words and numbers necessary for the general conduct of Navy business. Their usefulness depends on computing power, speed, and size. As with other critical resources, the Navy cannot depend on the private sector as the exclusive provider of the technology; it must be an active participant. This article focuses on computer circuits, and specifically on multiple-valued circuits, whose use improves computing power, speed, and size. We first consider binary circuits. Then, we discuss multiple-valued circuits. This serves as introduction to the main discussion, research at the Naval Postgraduate School on this important subject.

Disadvantages of Binary

Current computers are binary; they use two values of logic. For convenience, we represent these as 0 and 1. But 0 and 1 are simply symbols for some quantity, such as voltage, current, or charge. For example, in the most widely used circuits, 0 represents 0 volts and 1 represents 5 volts.

Binary is commonly used for historical reasons. Computers in the 1940's used relays, which had two stable states, open and closed. Tubes and transistors have two stable states, saturation (conducting) and cutoff (nonconducting). There are disadvantages to binary, however. One is evident by the number of bits needed to represent a number. For example, the decimal number 33 corresponds to 10001 in binary. That is, while five symbols are needed to represent 33 in binary, only two are needed in decimal. Unfortunately, the difference increases with the size of the number. This inherent complexity is hidden from most users by hardware (e.g., binary to decimal converters in digital

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **1993**	2. REPORT TYPE	3. DATES COVERED
4. TITLE AND SUBTITLE **Research on: Multiple-Valued Logic At The Naval Postgraduaet School**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Postgraduate School,Department of Electrical and Computer Engineering,Monterey,CA,93943**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The Navy's need for faster, smaller computers has in~ spired research on the use of more than two levels of logic in ? computer circuits. This paper focusses on work at NPS in multiple-valued logic. It discusses the development of multiple- valued programmable logic arrays (MVL-PLA), as well as computer-aided design tools for MVL-PLA's.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**8**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

watches) and by software (e.g., high level languages). However, certain disadvantages cannot be so simply hidden. Real penalties exist in both compactness and speed.

Compactness

The complexity of the circuits that perform addition, multiplication, and other arithmetic operations depends on the number of symbols used. For commonly used number systems, this complexity increases with the number of symbols. As a result, so does the chip area needed for devices. However, the most significant penalty is not with the devices, but the interconnect between devices. In VLSI circuits, about 70% of the chip area is used for interconnect, 20% for insulation, and 10% for devices. Interconnect corresponds to the chip area used to carry information around the circuit. Insulation is needed to separate devices and interconnect. With so much area used to connect devices, less area is available for logic.

The inefficient use of interconnect by binary also occurs outside the chip. Signals going into and out of a VLSI chip must use pins connecting the chip to the circuit board. This is called the *pinout problem.* That is, while significant reductions have occurred in the size of logic devices, there has been no comparable reduction in the size of integrated circuit pin connections. Strength and reliability dictate a (relatively large) minimum pin connection size. As a result, there is a need to better use existing pins.

Speed

A limit on the speed of arithmetic circuits is the carry (or borrow) between digits. For example, in binary addition, the most significant sum bit depends not only on the most significant bits of the two numbers being added, but also on all lower bits (contrast this to the least significant sum bit which does not depend on higher order bits). The dependence occurs through the carry between bits. Thus, computation of the most significant sum bit must wait until the carries out of all lower order bits are computed. Addition, when done in this way, cannot be performed at high speed.

However, there are multiple-valued number systems where this dependence does not exist. An example is the residue number system. In this system, operations that form the sum, difference, or product occur only at each digit independent of all other digits. Because of this independence, arithmetic operations are fast in the residue number system.

Multiple-Valued Logic

All of the disadvantages discussed above can be alleviated by using more than two levels of logic. The problem of a large number of bits needed to represent a number decreases as the number of logic values increases. Interconnect, both internal and external to the VLSI chip, is more efficiently used in multiple-valued logic. The carry propagation problem disappears with the proper choice of multiple-valued number system.

However, there remains the question of implementing a multiple-valued system. Modern computing devices naturally allow more than two levels of logic. For example, in charge-coupled devices (CCD), logic levels are represented as various quantities of charge[2]. Resonant-tunnelling devices[3] allow four or more levels of voltage. Fuzzy logic devices offer large numbers of logic levels; so many, that they are modelled as infinite; present circuit implementations exceed 15 levels[17]. Bio-molecular devices, where information processing occurs at the molecule level[6], operate on an enormous number of logic levels, conservatively estimated to be over 10,000.

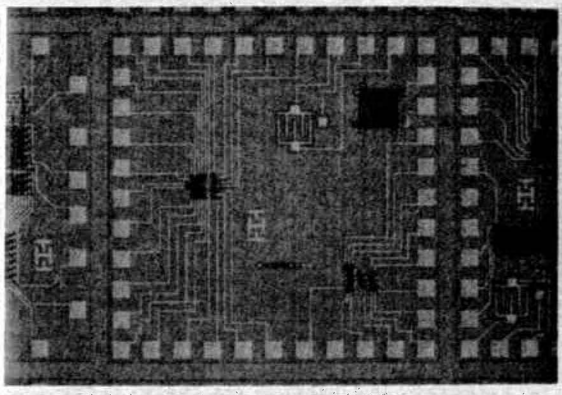
Multiple-Valued Programmable Logic Arrays

The design of logical circuits is a complex process. The vast amount of circuitry needed for even a simple operation like addition can be daunting. Because of this, a way is needed to organize circuits. The programmable logic array or PLA is a circuit structure designed to handle this complexity. PLA's are uniform and thus easily replicated in VLSI. They can be programmed, and therefore adapted to realize a given design specification. Because of this characteristic, our research at the Naval Postgraduate School has concentrated on PLA design.

In a cooperative venture with the University of Twente in Enschede, The Netherlands, we have developed a PLA circuit for charge-coupled device (CCD) technology (See Figure 1 from[2]). In CCD, values are stored in cells (capacitors) as charge. A logic 0, 1, 2, and 3 is stored as 0, 1,000,000, 2,000,000, and 3,000,000 electrons. Computation occurs as

Figure 1.

Photomicrograph of a Basic Charge Coupled Device Programmable Logic Array (Courtesy of the University of Twente, ©IEEE 1988).



charge moves among cells combining and breaking up. An advantage of CCD is compactness, more so than any other VLSI technology. While slower than the common CMOS technology, it is faster than disk and stands as a replacement for disk. The use of multiple-valued logic in CCD increases its density significantly. CCD is used as a light receptor, and the advent of logic circuits in CCD means that image processing can be done on the same chip in which the image is received. With large numbers of logic levels, there is no need for encoders between light receptors and the logic used to process and store the image. CCD circuits with many logic levels have been fabricated. For example, Hitachi has succeeded in building a 16 level CCD memory.

While the uniform structure of a PLA helps one to handle the high complexity of logic circuits, there is the problem of designing large PLA's. To make multiple-valued PLA design feasible for large systems, we initiated a program to develop computer-aided design tools.

Computer-Aided Design for Multiple-Valued Logic

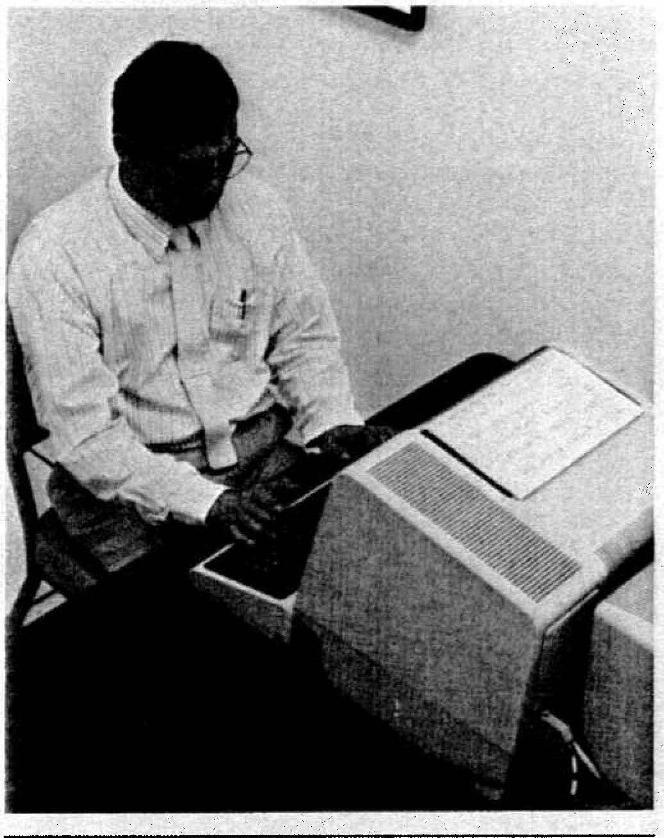
We began this program with a study of heuristic techniques for finding *minimal* PLA's, PLA's with the smallest structure. A heuristic will produce a design, although not necessarily a minimal one. In 1987 at the beginning of this study, there were three heuristic techniques. At that time, there was no systematic analysis of their relative merits; each existed separately. In an effort to gain an understanding of the multiple-valued PLA design process, we compared the three heuristics on the same functions (specifications). One interesting result[13] was that heuristics tended to perform better on different classes of functions, and there was a surprising advantage to applying all three and choosing the most compact realization.

We also provided a partial answer to an important question; How well do heuristics perform compared to the best that one could possibly do? To answer this, we developed the first practical approach to finding provably minimal solutions; an efficient search technique that is much faster than exhaustive search, but still guaranteeing a minimal solution. Indeed, without the improved search technique, we would have been unable to achieve the minimal solution because of the enormous computation time required by exhaustive search. This comparison provided much needed insight into heuristic methods and provided realism in our expectation of the quality of solutions provided by heuristic methods.

It is important to state that these experiments deal with small functions; for even medium size functions this experiment would have been impossible. For larger functions, the only approach is heuristic. Indeed, our research shows that often a minimal realization is *not* produced. Although a designer will not know how close his/her design is to the optimum, our study indicates that deviations are probably less than 10% away. Ours was the first systematic study of the problem.

Figure 2.

LTJG Ugur Ozkan Using HAMLET.



Our experience from this analysis showed the need for a tool to analyze PLA design heuristics. In 1988, we launched an extensive design project for such a tool. The result was HAMLET (Heuristic Analyzer for Multiple-Valued Logic Expression Translation), the first practical computer-aided (CAD) tool for multiple-valued logic. HAMLET[16] has the ability to analyze heuristics specified by the user, using either individual functions or randomly generated functions. This is a particularly useful feature. Instead of writing a separate program to generate test inputs, this is done automatically in HAMLET. Of special significance is HAMLET's ability to collect and automatically display heuristics. Tasks that normally took man-days can now be literally done in keystrokes. HAMLET can produce ensembles of test cases whose statistical properties can be controlled. The search algorithm used in reference 13 was also incorporated into HAMLET. We are now able to use the search to find solutions that were impossible to obtain with any previous method. The search can be controlled. At one extreme, it can go directly to a PLA realization of a given function. At the other, it can perform a complete search, producing a known minimal solution. In effect, this is like a dial, one end of which corresponds to fast but poor solutions, while the other end corresponds to slow but minimal solutions. HAMLET can also do design, producing the layout of a PLA, given a specification. Figure 3 shows one PLA layout produced by HAMLET. HAMLET is designed to have a long lifetime. It is easily modified; indeed an improved heuristic has been added[14], as well as an entirely new approach[4] (see below). Also, HAMLET has been used by researchers in at least four countries.

Simulated Annealing

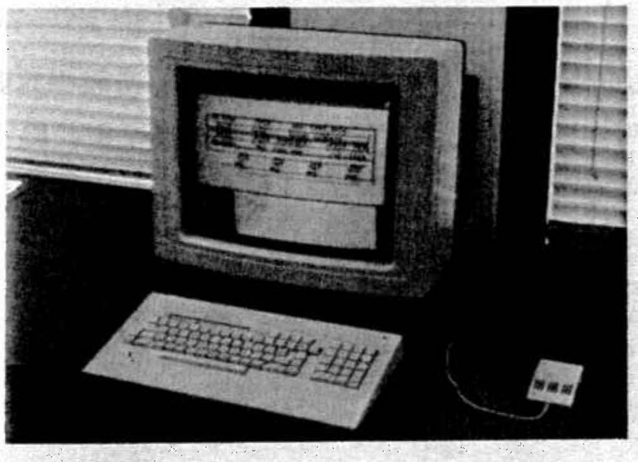
When metal or glass is heated and then slowly cooled, it is said to undergo *annealing*. The final cooled state is similar to a crystal and represents a low energy state. If it is cooled at a high rate, atoms have less of a chance to align and a higher energy state occurs.

In 1953, Metropolis[11] proposed a method of optimizing complex problems that mimics this process. It is called *simulated annealing*. We have formulated simulated annealing for the PLA minimization problem as follows. One can satisfy a given PLA specification by producing a set of implicants. Each implicant represents a part of the whole PLA. Solving the PLA minimization problem, therefore, is equivalent to finding the fewest implicants that satisfies the specification. Indeed, there is a direct relationship between the number of implicants and the chip area occupied. Thus, reducing the number of implicants by half, for example, results in almost a one-half reduction in chip area. Implicants can be manipulated. For example, one implicant can be divided into two, pairs sometimes can be combined into a single implicant, and pairs can sometimes be changed into another pair, or a triple, etc.. Ideally, one would like to combine a pair of implicants into one, since this reduces PLA chip area. However, in a given set of implicants, it is quite possible that no pairs combine. Such a set is said to be a *local* minimum. However, by replacing a pair by another pair, or by a triple, etc., it is possible to create a new set of implicants, where now many combine, leading to a smaller set. The descriptor "local" means there are no implicant sets nearby that contain fewer elements. It is not a "global" minima if there is another set of implicants (somewhere) that has fewer elements.

The derivation of a minimal set is a difficult problem. A heuristic generates a solution to a problem, but not necessarily a minimal one. The PLA minimization problem belongs to the set of NP-complete problems, which means that the best known algorithm for finding the minimal solution increases exponentially in complexity as the problem size increases. It is for this reason that there is so much interest in heuristics. Simulated annealing has the benefit that, under certain conditions, a minimal solution will be found with a probability that approaches 100% as the process continues. This is unlike any other known heuristic for PLA minimization, where, for certain PLA specifications, the probability of finding a minimal solution is 0%.

Figure 3.

PLA Layout Produced by HAMLET.

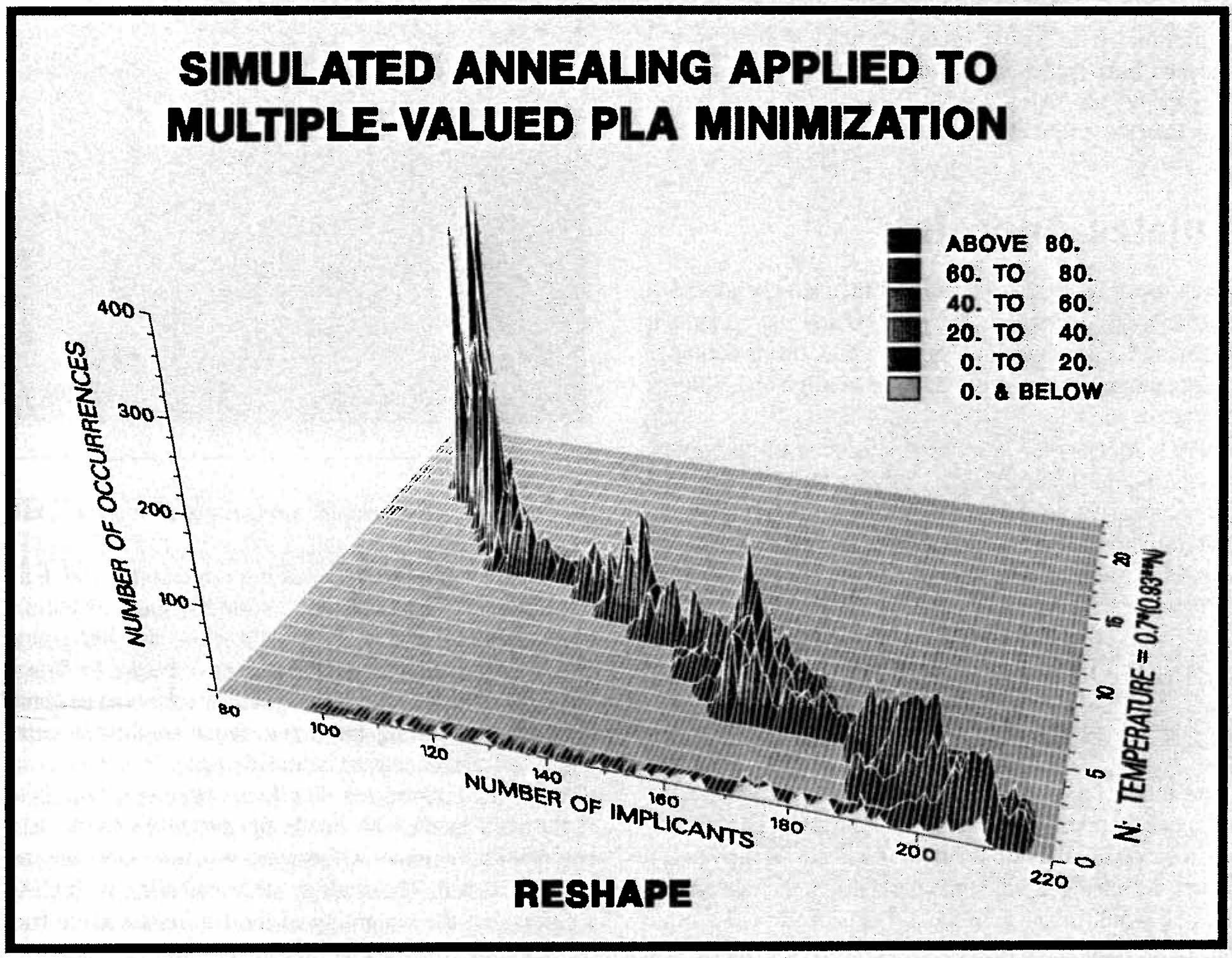


In effect, simulated annealing is a search through a space of solutions. The basic computation is a *move*. That is, given one solution to the PLA specification, another solution is obtained by making a move. Each move begins by first selecting a pair of implicants. If the two implicants can be combined, they are, completing the move. If the implicants cannot be combined, a replacement is considered with the fewest number of equivalent implicants. In effect, a weighted coin is tossed. If the coin lands with heads up, the replacement is made, completing the move. Otherwise, it is not made, and another pair is selected. The analogy with annealing is this. At each temperature, the weighting of the coin is the same for some fixed number of moves. In our programs, we allow 20,000 moves at each temperature. At high temperatures, the weight is such that heads almost always comes up, which means that all moves are accepted. However, at low temperatures, the weight is shifted to tails, so that fewer moves are accepted which increase the number of implicants. At any temperature, a move that reduces the number of implicants is accepted. Thus, at low temperatures, the mixture of generated moves is predominantly PLA complexity-decreasing moves. The existence of complexity-increasing moves at all temperatures is important; it allows the system to escape from local minima. This is analogous to annealing, in which atoms at high temperature move about freely. At lower temperatures the freedom is restricted, as they settle into a low energy state.

Figure 4 shows how simulated annealing performs on a multiple-valued function that begins with 96 implicants. This solution was obtained by applying a heuristic to some given PLA specification. The axis labeled "number of product terms" measures the quality of the solution to the PLA specification; the fewer the better. The axis marked temperature

Figure 4.

Simulated Annealing Applied to Multiple-Valued PLA Minimization.



measures the probability with which implicant increasing moves are accepted. At higher temperatures (toward the front), almost all implicant-increasing moves are accepted. The vertical axis represents the number of times a solution is found at the temperature and the number of implicants on the x-y plane. Here, color encodes the vertical deviation. In effect, the vertical axis shows how the simulated annealing program is doing with respect to the number of product terms. Specifically, at the highest temperature (the slice closest to the front), one can see how the simulated annealing progresses from the initial solution of 96 implicants up to as many as 216 implicants. This is described as *melting*. As the temperature decreases from this point, there is steady progress towards PLA specifications with fewer implicants. In this application of simulated annealing, a solution of 88 implicants is achieved. The behavior of the implicants is similar to the behavior of individual atoms as metal or glass anneals; as the temperature decreases, the form of the final material takes shape as individual atoms occupy a site in the structure corresponding to low energy.

A comparison of simulated annealing with other PLA minimization heuristics shows that it is superior on an average case basis. That is, using HAMLET, random functions were generated, heuristics were applied to all, and the results compared. Simulated annealing surpasses the all known heuristics on the solutions obtained. Although, more time is needed, simulated annealing has the benefit that its performance can be controlled. That is, with larger computation times, one can achieve a more compact realization.

Conclusions

The Navy has a tradition of commitment to computing, as epitomized by the contributions of Rear Adm. Grace Murray Hopper. Indeed, in the past 50 years, significant improvement has occurred in Navy systems because of the computer. It is clear that the next 50 years will bring even further improvement. The promise of compact, high speed computers has inspired our work on multiple-valued logic.

Multiple-valued logic offers a means to overcome significant disadvantages of binary. It makes more efficient use of chip area by encoding more information in the wires that connect devices, and it allows the use of high-speed carry-less operations. Our work at the Naval Postgraduate School has been focused on development of techniques (PLA's) and the tools (CAD) needed to make multiple-valued logic a reality.

Our current effort focuses on expanding simulated annealing. This includes a promising approach to high-speed computations; indeed, initial indications show a modification of the conventional simulated annealing paradigm can produce computation speeds higher than any known heuristic, while maintaining almost the same capability to minimize chip area. We are also pursuing the use of parallel computers to design PLA's[15], including simulated annealing.

Acknowledgements

The author gratefully acknowledges the cooperation of the Naval Research Laboratory in this research. Support for this work has also been provided by the Naval Postgraduate School Direct Funded Research Program, the National Science Foundation, the National Research Council and the North Atlantic Treaty Organization. Special thanks are due the many Naval Postgraduate School students who have contributed to this project, including, LT Hoon-seop Lee, MAJ Yong Ha Ko, LTJG Ugar Ozkan, CAPT Alan W. Watts, LT Yao-Ming Wang, LT S. Onur Oral, LT R. Charles Earle, LT Cem Yilderm, and LT Dave York. Special thanks are also due LCDR John Yurchak, the architect of HAMLET, Professor Chyan Yang, Co-Principal Investigator, and Professor Gerhard Dueck, National Research Council Postdoctoral Associate and collaborator on the simulated annealing research.

Biography

Jon T. Butler is Professor of Electrical and Computer Engineering at the Naval Postgraduate School. He teaches computer design. His research interests include multiple-valued logic and fault tolerant computing. He is a Fellow of the IEEE. He has been Editor of the *IEEE Transactions on Computers* and the *Computer Society Press*, and is the past Editor-in-Chief of *Computer*. Presently, he is the Editor-in-Chief of the *Computer Society Press*. Professor Butler is a co-recipient of the Award of Excellence and the Outstanding Contributed Paper Award for papers presented at the International Symposium on Multiple-Valued Logic in 1985 and 1986, respectively. Currently, he is a member of the Board of Governors of the IEEE Computer Society.

References

1. Besslich, P. W., "Heuristic minimization of MVL functions: A direct cover approach," *IEEE Trans. on Comput.*, February 1986, pp. 134-144.
2. Butler J. T., and H. G. Kerkhoff, "Multiple-valued CCD circuits," *Computer*, Vol. 21, No. 24, pp. 58-69, April 1988.
3. Capasso, F. et al, "Quantum functional devices: resonant tunneling transistors, circuits with reduced complexity, and multiple-valued logic," *IEEE Trans. on Electron. Devices*, Oct. 1989, pp. 2065-2082.
4. Dueck, G. W., R. C. Earle, P. P. Tirumalai, and J. T. Butler, "Multiple-valued programmable logic array minimization by simulated annealing," *Proc. of the 22nd Inter. Symp. on Multiple-Valued Logic*, May 1992, pp. 66-74.
5. Dueck, G. W., and D. M. Miller, "A direct cover MVL minimization using the truncated sum," *Proc. of the 17th Inter. Symp. on Multiple-Valued Logic*, May 1987, pp. 221-227.
6. Kameyama, M., and T. Higuchi, "Prospects of multiple-valued bio-information processing systems," *Proc. of the 18th Inter. Symp. on Multiple-Valued Logic*, May 1988, pp. 360-367.
7. Kameyama, M., T. Sekibe, and T. Higuchi, "Highly parallel arithmetic chip based on multiple-valued bi-directional current-mode logic," *IEEE Jour. of Solid-State Circuits*, Vol. 24, No. 5, Oct. 1989, pp. 1404-1411.
8. Kawahito, S., et al, "A high-speed compact multiplier based on multiple-valued bi-directional current-mode circuits," *Proc. of the 17th Inter. Symp. on Multiple-Valued Logic*, May 1987, pp. 172-180.
9. Kerkhoff, H. G., and J. T. Butler, "Design of a high-radix programmable logic array using profiled peristaltic charge-coupled devices," *Proceedings of the 16th International Symposium on Multiple-Valued Logic*, May 1986, pp. 100-103.
10. Kirkpatrick, S., C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, No. 4598, 13 May 1983, pp. 671-680.
11. Metropolis, N., A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, *Jour. of Chem. Phys.*, 21, 1953, pp. 1087.
12. Pomper, G., and J. A. Armstrong, "Representation of multivalued functions using the direct cover method," *IEEE Trans. on Comput.* Sept. 1981, pp. 674-679.
13. Tirumalai, P. P., and J. T. Butler, "Minimization algorithms for multiple-valued programmable logic arrays," *IEEE Trans. on Computers*, Feb. 1991, pp. 167-177.
14. Yang, C. and Y.-M. Wang, "A neighborhood decoupling algorithm for truncated sum minimization," *Proceedings of the 20th International Symposium on Multiple-Valued Logic*, May 1990, pp. 153-160, pp. 75-82.
15. Yang, C. and O. Oral, "Experiences of parallel processing with direct cover algorithms form multiple-valued mini-

mization," *Proceedings of the 22nd International Symposium on Multiple-Valued Logic*, May 1992.

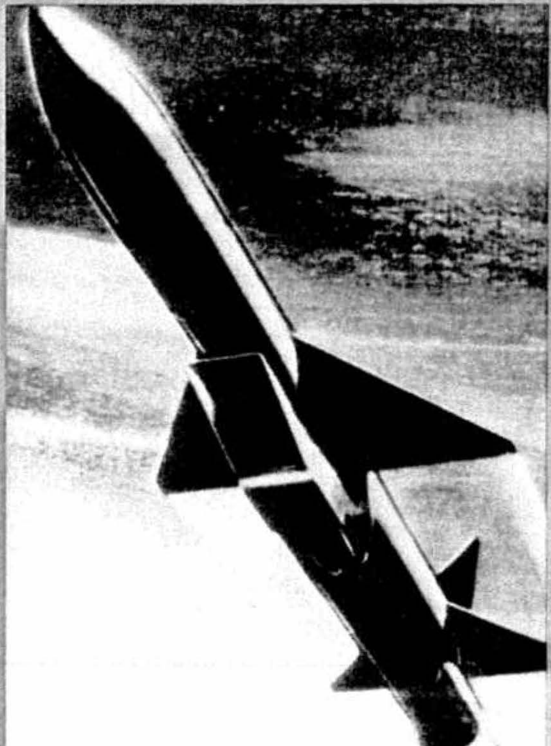
16. Yurchak, J. M. and J. T. Butler, "HAMLET – An expression compiler/optimizer for the implementation of heuristics to minimize multiple-valued programmable logic arrays", *Proceedings of the 20th International Symposium on Multiple-Valued Logic*, May 1990, pp. 144-152, For an expanded version, see Yurchak, J. M. and J. T. Butler, "HAMLET user reference manual," Naval Postgraduate School Technical Report NPS-6290-015, July 1990.

17. Zadeh, L. "Fuzzy logic," *Computer*, Vol. 21, No. 24, pp. 83-93, April 1988.

Naval Research Reviews

Office of Naval Research
Four / 1992 One / 1993
Vol XLIV / Vol XLV

Mixing Enchancement
for Air-Breathing Propulsion



Non-Axisymmetric Jets
Increase Mixing